Software Engineering Department

Braude College

Capstone Project Phase A – 61998

Detection of Rare Coins

25-1-R-14

By Hassan Said Ahmad 211920624 Hassan.Said.Ahmad@e.braude.ac.il

Lev Leyfer 321385064 Lev.Leyfer@e.braude.ac.il

Supervisor: Zeev Frenkel

Abstract

The detection of rare coins presents a unique challenge in the field of object recognition, particularly when distinguishing between common and rare variations based on subtle differences such as minting year, inscriptions, or design variations. This project focuses on developing an AI-driven solution for the detection of rare Israeli coins by date, utilizing advanced computer vision techniques and machine learning algorithms. By leveraging a trained YOLO model, the system is designed to identify coins accurately in real-time, either from static images or live video input. The primary goal is to provide numismatists with an efficient tool for coin classification, reducing the risk of human error and making the process faster and easier. The system evaluates coins based on high-resolution image input, detecting key features and minting details that distinguish rare coins from common ones.

1. Introduction

The field of numismatics, or coin collecting, has long relied on expert analysis to identify rare coins, with rarity often determined by factors such as mint year, production errors, or limited circulation. However, this manual identification process can be time-consuming, prone to error, and inaccessible to those without specialized knowledge. Recent advancements in artificial intelligence (AI) and computer vision offer a powerful solution to these challenges by automating the recognition and classification of rare coins. Existing solutions work with images of single coins expected to be rare. Nevertheless, many non-professional coin collections are composed from coins presented in circulations or replaced from circulations in the last 100 years. To compose such collections, numismatics manually coin by coin explore placers of coins presented in various cashboxes. For such collections, existing rare coin identification solutions are inconvenient to use.

This project focuses on developing an AI-based system for detecting rare Israeli coins out of placers of coins, with a specific emphasis on identifying distinctions based on minting dates. By training a machine learning model, particularly leveraging the YOLO (You Only Look Once) object detection framework, our system can accurately identify and differentiate rare coins from common ones in real time using either images or live video input.

1.1 Methods

Our application uses the following methods:

1. **Data Collection and Annotation:**
    - Making a dataset of 100 images of agorot (Fig. 1)
    - Annotate the images to mark its year stamps (Fig. 1)

2. **Preprocessing:**
    - Resize, normalize, and augment the dataset using techniques such as rotation, scaling, and contrast adjustments to simulate real-world variations.
    - Implement image segmentation to separate coins from backgrounds for better focus during training.
3. **Model Training:**
    - Use the YOLO (You Only Look Once) object detection framework for training a model capable of detecting coins and their features in real time.
    - Fine-tune the pre-trained YOLO weights with the annotated dataset to enhance model accuracy.
4. **Evaluation and Testing:**
    - Evaluate the model using metrics such as precision, recall, and F1 score.
    - Conduct validation on unseen data to measure generalization capability.
    - Implement real-time testing using video feed or camera input.

Figure 1. Coin with year stamp. On the left we can see an image of a 10 agorot coin with the hebrew letters on the bottom (התשע"ד) which corresponds to the gregorian year of 2013/2014 and on the right is (התשע"ה) which corresponds to the gregorian year of 2014/2015.

1.2 Technologies

Our application is based on the following technologies:

1. **YOLO Object Detection Framework:**
    - Reason: YOLO is known for its real-time performance and high accuracy in object detection tasks. Its ability to process images quickly makes it ideal for detecting coins in live video streams or batch image analysis.
2. **Python:**
    - Reason: Python offers a wide range of libraries for machine learning (we shall be using PyTorch) and image processing . It is also compatible with YOLO integrations and facilitates rapid prototyping.
3. **PyTorch:**
    - Reason: This deep learning framework provides tools for training and deploying models, including support for YOLO implementation and model optimization.
4. **LabelImg:**
    - Reason: LabelImg is an annotation tool that simplifies the process of creating labeled datasets required for training the object detection model.
5. **Hardware:**
    - NVIDIA GPUs: Essential for accelerating the training of deep learning models.

1.3 Tools

Our application utilizes the following tools:

1. **Dataset Tools:**
    - Roboflow for preprocessing and labeling dataset.
2. **Development Tools:**
    - Git for version control and collaboration.
3. **Visualization Tools:**
    - Matplotlib and Seaborn for analyzing model performance through visualizations.

1.4 Objectives

The primary objective of this project is to develop a tool that helps numismatics quickly and accurately identify rare coins. By leveraging modern technology, we aim to streamline the process of coin detection, making the identification of rare coins more accessible and efficient for a wide range of users

2. Background and Related Works

The use of artificial intelligence and machine learning in object detection and classification has grown significantly in recent years, with applications spanning various domains such as facial recognition, autonomous vehicles, and medical diagnostics. Within the field of numismatics, AI-driven systems are emerging as a promising tool for automating the identification and authentication of coins, though research in this area is still developing [1].

Early approaches [1] primarily relied on traditional image processing techniques such as edge detection and template matching. However, these methods often struggled with variations in lighting, orientation, and wear on the coins, leading to inconsistent results[3]. More recently, deep learning models such as Convolutional Neural Networks (CNNs) and object detection frameworks like YOLO and Faster R-CNN have demonstrated superior performance in recognizing objects with high precision [4].

Now we will briefly review two most popular existing related tools and algorithms.

2.1 Coinoscope

Coinoscope [6] is an Android app developed by Miccron that serves as a visual search engine for coins. This app allows users to identify coins by taking a photo with their smartphone camera. It then compares the image to a vast database of coins, showing a list of similar matches. However, while Coinoscope can provide a range of possible years for a coin, it does not identify the exact year of the coin, which means users are given a general time frame rather than a precise date.

In addition to its core functionality of coin identification, Coinoscope offers users an option to tap on any coin in the results to view more detailed information through their phone's web browser, including an estimate of its value. Coinoscope provides only a general time frame for coins, which may not meet the needs of users requiring exact mint years. Its accuracy depends on the quality and size of its database, limiting effectiveness for rare or obscure coins. Additionally, it is constrained to mobile platforms, which may not suit professional numismatists preferring desktop tools for detailed analyses.

2.2 Numista

Numista [5] is an online platform and database for coin collectors that offers detailed information on a vast range of coins from around the world. Unlike Coinoscope, which uses a mobile-first image recognition approach, Numista relies on a combination of textual search and advanced filtering options based on attributes such as material, diameter, and mint marks. Users can manually input details about a coin to find matching results and access extensive metadata, including the year of minting, circulation status, and historical context which can be time-consuming and challenging for those unfamiliar with a coin's details.

2.3 Hebrew written year vs. Gregorian numerical year

Our application introduces AI-powered image recognition to identify coins and extract their exact mint year, surpassing the existing tools by combining precision, automation, and real-time functionality. In Hebrew, letters are used as numbers, a system known as Gematria[2]. Each letter has a numerical value, and when letters are combined, their values are added together to represent numbers, including years in the Hebrew calendar. Here's how the process works[2]:

Hebrew Letters and Their Numerical Values:

- The first nine letters represent the numbers 1 through 9:
  - א = 1, ב = 2, ג = 3, ד = 4, ה = 5, ו = 6, ז = 7, ח = 8, ט = 9
- The next nine letters represent the tens:
  - י = 10, כ = 20, ל = 30, מ = 40, נ = 50, ס = 60, ע = 70, פ = 80, צ = 90
- The remaining letters represent the hundreds:
  - ק = 100, ר = 200, ש = 300, ת = 400

Years in the Hebrew Calendar:

The Hebrew calendar year is typically written using letters that correspond to its numerical value, omitting the thousands place. For example:

- **5784** (the current Hebrew year) is written as **תשפ"ד**.
  - 784 = **(4) ד** + **(80) פ** + **(300) ש** + **(400) ת**
  - The "thousands" digit (5 in 5784) is implied by context and not written.

Converting Letters to Numbers:

1. Break down the Hebrew letters into individual characters.
2. Assign the numerical value of each letter based on the Gematria system.
3. Add up the values to get the year’s numerical representation.

Example: תשפ"ד

- 400 = ת
- 300 = ש
- 80 = פ
- 4 = ד

Total: 400+300+80+4=784. The thousands (5) are implied, so the year is 5784.

And now to make it into a civil year , we omit the first digit , and then we add 1240 , in our example is 784 + 1240 = 2024

The Hebrew calendar counts years starting from the creation of the world, traditionally calculated as 3761 BCE. The Gregorian calendar counts years from the presumed birth of Jesus Christ. The difference between the two systems is 3761 years.

- If you're after **Rosh Hashanah**, you use the full difference (3761).
- If you're before **Rosh Hashanah**, the Hebrew year corresponds to the Gregorian year minus 1, since the Hebrew year hasn't yet started its next cycle.

3. Expected Achievements

3.1 Outcomes

The outcomes of our project aim to accurately and efficiently make a tool to assist in identifying rare Israeli coins. We intend to develop a system that provides collectors and enthusiasts with an intuitive platform for coin detection and classification. The expected outcomes include:

1. **Accurate Detection and Classification:**
    - The system will accurately identify Israeli coins based on mint year.
    - It will differentiate between rare and common coins with a high degree of precision.
2. **Real-Time Functionality:**
    - The tool will support real-time detection through live video feeds or static images, allowing users to evaluate coins instantly.
3. **Analysis:**
    - Users will benefit from insights derived from correlations between coin attributes and their historical rarity.

By achieving these outcomes, the project will enhance the accessibility and accuracy of coin authentication, supporting both professional collectors and casual enthusiasts.

3.2 Unique Features

1. YOLO-Based Coin Detection

A core feature of the project is the integration of the YOLO [7] object detection framework. YOLO's speed and accuracy make it ideal for analyzing high-resolution coin images and live video streams. Its ability to identify fine-grained details, such as mint years and inscriptions, ensures precise detection critical for this application.

2. Custom Dataset for Rare Israeli Coins

The system will be trained using a specialized dataset of high-resolution images of Israeli coins. This dataset will be made using up to 100 common coins that we will be making ourselves.

3. Real-Time Video Analysis

In addition to static image uploads, the system will support live video input for real-time coin detection. This feature enhances usability and provides instantaneous feedback to users.

3.3 Criteria for Success

1. **Accuracy and Reliability:**
    - The detection model must achieve a high degree of accuracy in identifying rare coins and their unique features.
2. **Optimized Performance:**
    - The application must perform efficiently with steady framerates, ensuring smooth real-time video analysis.
3. **User Friendly assistance:**
    - The application must simply alert on the presence of interesting coins in the placer of coins. For detected interesting coins, the application must point on the placement of these coins in the placer and provide information why these coins can be interesting for the user.

By meeting these criteria, this project will deliver a valuable tool for the numismatic community, modernizing and simplifying the process of rare coin detection and classification.

4. The Process

4.1 General research

4.1.1 Objective

Our research focused on exploring the world of rare coin detection to help design a reliable and efficient object detection system for coins. We set out to answer these key questions:

1. **Characteristics of Rare Coins:** What are the unique features that distinguish rare coins from common ones? In our project identifying the rarity of coins based on their year of minting is the primary goal.
2. **Target Use Cases:** Who are the primary users of rare coin detection systems (e.g., collectors, cashbox holders, appraisers, or museums)?
3. **Challenges in Detection:** What factors complicate the accurate detection and classification of rare coins, such as wear and tear, dirt and rust, or environmental lighting and angles of rotation?
4. **Existing Solutions:** What detection systems currently exist for identifying rare coins, and what are their limitations?

To address these questions and build our approach, we explored various resources, including academic papers, videos, and technical blogs. After consolidating our findings, we discussed key focus areas for developing our detection system.

4.1.2 Key Insights and Goals

- **Unique Coin Features:** Rare coins often have intricate designs and historical significance. These attributes need high-resolution imaging and robust feature extraction techniques.
- **Target Users:** Collectors, appraisers and cashbox holding enthusiasts require tools that are both accurate and user-friendly.
- **Challenges:** Wear, discoloration, and inconsistent lighting conditions can hinder detection accuracy.

4.1.3 Constraints and Challenges – Rare Coin Detection

1. Variability in Coin Appearance
Coins vary significantly in size, color, and wear. Similar coins have varying year dates that need identifying.

2. Environmental Factors
Lighting and background interference can distort the detection process. To mitigate this, pre-processing techniques like histogram equalization and background subtraction will be explored.

3. Dataset Limitations
For identifying Israeli coins based on their year of minting. The dataset should include high-resolution images of coins, annotated with details such as the year of minting. This is crucial, as coins from specific years or limited production runs are often of particular historical and numismatic significance.

The Hebrew calendar year inscriptions on coins, which differ from the Gregorian calendar, add a layer of complexity in identifying mint years accurately.

Creating such a dataset is foundational to developing a reliable object detection system for rare Israeli coins. By addressing the challenges above and ensuring high-quality, annotated data, the project can achieve its purpose by identifying coins by year, even in challenging conditions such as wear or poor lighting.

4. Real-Time Detection
The detection system must balance accuracy with speed, especially for live video input.

4.2 Object Detection and YOLO

4.2.1 Objective

Our research on object detection algorithms, specifically YOLO (You Only Look Once) [7], focused on leveraging its efficiency and accuracy for real-time rare coin detection.

4.2.2 Key Research Points

1. **YOLO Advantages:** YOLO processes images in a single pass, making it suitable for real-time applications.
2. **Pre-Processing:** Understanding how to prepare training datasets, including annotation formats to improve detection accuracy.
3. **Model Training:** Investigating techniques for training YOLO on custom datasets.
4. **Post-Processing:** Learning about methods to refine detection results, such as non-maximum suppression (NMS) [8] (YOLO uses this as a part of its process).

4.2.3 Constraints and Challenges

1. Hardware Limitations
Training YOLO models can be computationally expensive, requiring high-performance GPUs. Real-time detection on low-powered devices may require additional optimization techniques.

2. Dataset Quality
The model's performance is heavily dependent on the quality and diversity of the dataset. Rare coins with inconsistent or limited examples in the dataset may result in reduced accuracy.

3. False Positives/Negatives
Given the intricate nature of rare coins, the model may mistakenly classify common coins as rare (false positives) or fail to detect rare coins entirely (false negatives).

4.3 Methodology and Development Process

We adopted the Agile methodology [9] to iteratively develop the rare coin detection system. This approach allows flexibility and rapid adaptation based on user feedback. The development is divided into the following phases:

1. **Data Collection and Preprocessing**
    - Collect high-quality images of 10 agorot coins, we will gather around 100 coins images of this specific coin denomination.
    - Label datasets using annotation tools.
    - Apply pre-processing techniques like resizing, normalization, rotation.
2. **YOLO Model Training**
    - Train a custom YOLO model using the annotated dataset.
    - Evaluate performance using metrics like mAP (mean Average Precision) [10].
3. **Integration with Camera Systems**
    - Develop a pipeline to process live video or photo inputs.
    - Implement real-time detection using the trained YOLO model.
4. **Result Refinement and Database Integration**
    - Cross-reference detected coins with a database of rare coins to validate accuracy.
    - Display detailed analysis, including coin features and historical data.
5. **Testing and Validation**
    - Test the system on diverse lighting and background conditions.

6. **GUI**
    - A simple GUI interface would be implemented.
    - A button to start video mode.
    - A button to capture a photo and a button to process the given photo.
    - There will be a text box with the information after processing the coin.
    - In addition to a text box that tells the algorithm which years are needed.
    - A slider which gives the option to tell the algorithm the allowed accuracy of detection.

4.4 Evaluation Criteria

- **Accuracy:** Detection and classification of rare coins with minimal false positives/negatives.
- **Speed:** Real-time processing of live video input.
- **Scalability:** The ability to handle an expanding database of rare coins.

5. Product

5.1 Requirements

Functional:

1	The system shall allow users to take a picture or live video streams for coin detection.
2	The system shall manage and process coin image data in real-time.
3	The system shall store data about the coins in the database
4	The system shall support dynamic detection modes, switching between live video input and static image uploads.
5	The system shall classify detected coins based on rarity, mint year, and origin.
6	The system shall provide detailed information about the detected coin such as an estimated value and any additional information that we have in the database.
7	The system shall provide a confidence score for each detected coin.
8	The system shall handle batch processing of multiple images from a directory specified in the command line.
9	The system shall utilize efficient algorithms and pre-processing techniques to minimize computational load.
10	The system shall provide visually accurate representations of detected coins, highlighting distinct features (date of coin).

Non-functional:

1	The processing speed for a static image shall be 2 seconds or less.
2	Handling live video streams with a frame processing time of 200 milliseconds or less.
3	The coin detection and classification accuracy shall be at least 80% for known coin types.
4	Support batch processing of up to 100 images in a single run without performance degradation.
5	Having an operational reliability of 99.9% during continuous processing tasks.
6	There will be support for common image file formats (e.g., JPEG, PNG) for static image uploads.
7	The architecture shall allow easy replacement or updating of the detection model and coin database without requiring changes to the core logic.
8	Startup time for the application must not exceed 5 seconds on a system with SSD storage.
9	Detection models must be trained to handle environmental variances such as different lighting conditions with less than a 10% drop in accuracy.

5.2 Architecture Overview

This architecture is designed to optimize the coin detection process, focusing on performance, flexibility, and scalability, while accommodating the command-line interface (CLI) and a simple graphical user interface (GUI). It balances the user's need for accurate detection and real-time processing.

5.2.1 User Interaction Layer

Purpose: Provides users with tools to interact with the system via a command-line interface (CLI) and a simple GUI.

Components:

Command-Line Interface (CLI):

- Allows users to upload coin images or start live video streams by specifying file paths or video sources in the terminal.
- Displays analysis results in the console, including coin rarity and other relevant data.
- Supports batch processing of images from a specified directory via command-line flags.

Simple GUI:

- **Image Input:** Provides a simple interface where users can select and upload coin images.
- **Video Input:** Offers a button to initiate live video streams for real-time coin detection.
- **Basic Control:** Enables users to start/stop video detection and manually load image files using a basic interface with clear labels.
- **Display Results**: Shows basic coin details such as classification and estimated value in a simple, readable format.
- **User Input:** A slider to allow the user to pick required accuracy and a textbox that allows the user to add specific years.

5.2.2 Data Collection and Preprocessing Layer

Purpose: Manages incoming images/videos and prepares them for model inference.

Components:

Image/Video Preprocessing Module: Resizes, normalizes, and enhances input images for optimal model performance in addition to applying background subtraction and lighting corrections for consistent results.

Data Augmentation (Training Phase):Generates variations of training images to improve model robustness (e.g., rotations, brightness changes).

5.2.3 Detection and Classification Layer

Purpose: Executes the core functionality of identifying and classifying rare coins.

Components:

- **YOLO Model**: A pre-trained and fine-tuned object detection model tailored to detect coin attributes like mint year and what coin is it and then outputs bounding boxes, confidence scores, and classifications in real time.
- **Feature Extraction Module**: Extracts detailed attributes (e.g., mint year) from the detected coins to use our algorithm to convert to actual civil year instead of the hebrew year.

5.2.4 Data Analysis and Insights Layer

Purpose: Processes detection results to generate meaningful insights for the user.

Components:

- **Coin Rarity and Value Estimation Module**: Correlates detected features with database information to determine rarity and historical significance.
- **Visualization Module**: Presents results in a clear format, including annotations.

5.2.5 Storage and Database Layer

Purpose: Maintains information about the coins if there is a need.

Coin Database: Stores metadata for known coins.

5.2.6 Data Flow

1. **Input**: The user captures an image or starts a video stream via the CLI or GUI.
2. **Pre-processing**: The image or video stream is pre-processed to optimize the data for detection (e.g., resizing).
3. **Coin Detection**: The system applies the trained detection model to find coins in the image or video stream and then the system finds the Hebrew letters stamped on the coin.
4. **Classification & Extraction**: Detected coins are classified based on known characteristics such as rarity, mint year, and any additional metadata is fetched from the database , the mint year is extracted by translating the Hebrew letter date into a civil year according to the algorithm of conversion that was stated previously.
5. **Output**: Results are displayed via the console or through the GUI .

5.2.7 System Requirements

To train the model you would need -

- **Hardware**: The system can run on a typical machine with a multi-core processor, at least 8 GB of RAM, and a reasonable GPU for training and inference tasks. A dedicated GPU (e.g., NVIDIA GTX) is recommended for real-time video processing.
- **Software**: The system will be implemented using Python, with PyTorch for model training and inference. It will also use open-source libraries for image processing (e.g., OpenCV) and file handling.

To run the model you would not need many resources , you can even run the model on most smartphones locally , almost all of the models that we use are android friendly [11] (android os is the most popular in the world).

5.2.8. Summary

This architecture is designed to meet the needs of researchers and enthusiasts interested in coin detection, providing both flexibility (CLI and GUI options) and high performance (real-time video and batch processing). The modular approach ensures that the system is scalable and adaptable, allowing for easy updates to the detection model and database. This architecture also optimizes resource usage, balancing speed, accuracy, and computational load.

5.3 Use case

Use Case

The following Use Case diagram shows both the user and his interaction system

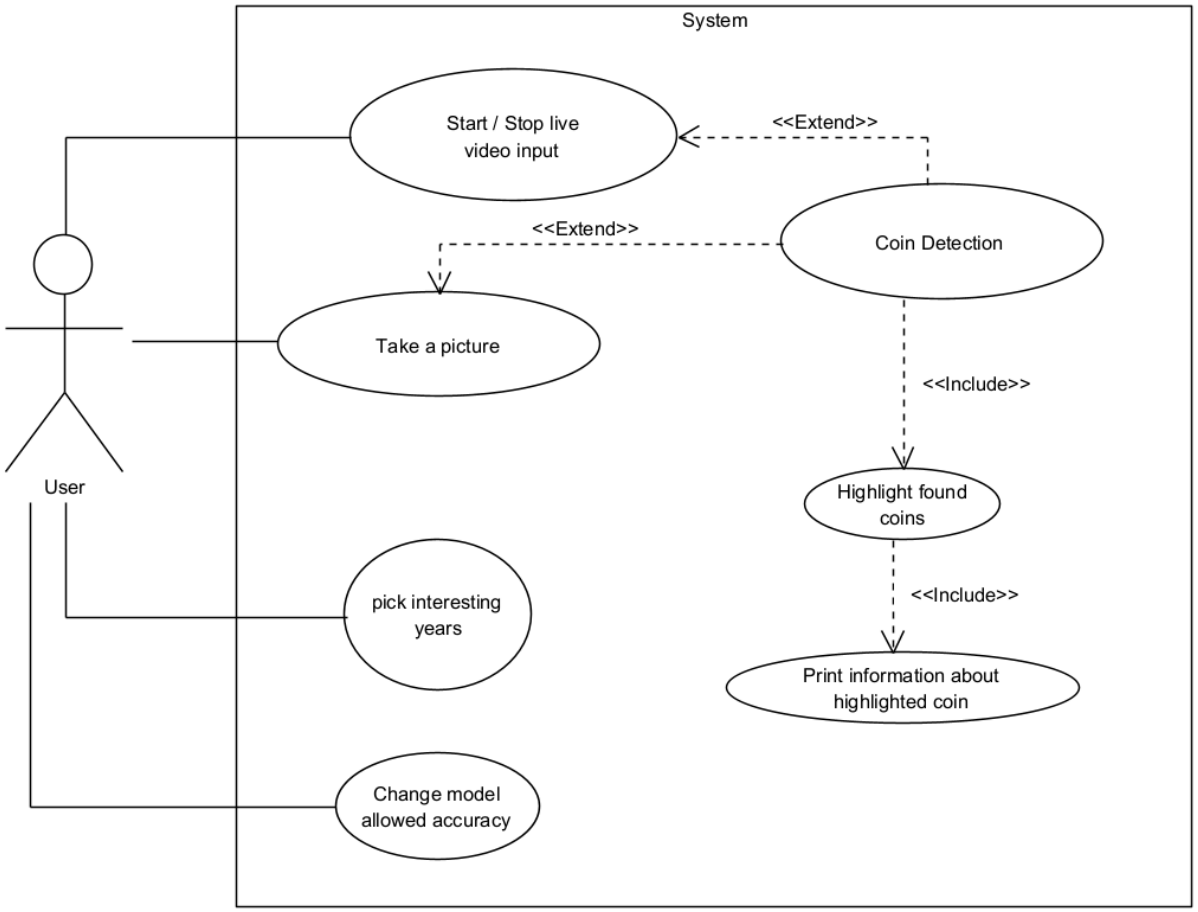

Figure 2: Use Case diagram

5.4 Activity Diagram

The following activity diagram depicts the user's interaction with the system and GUI, outlining the flow of activities and decision points

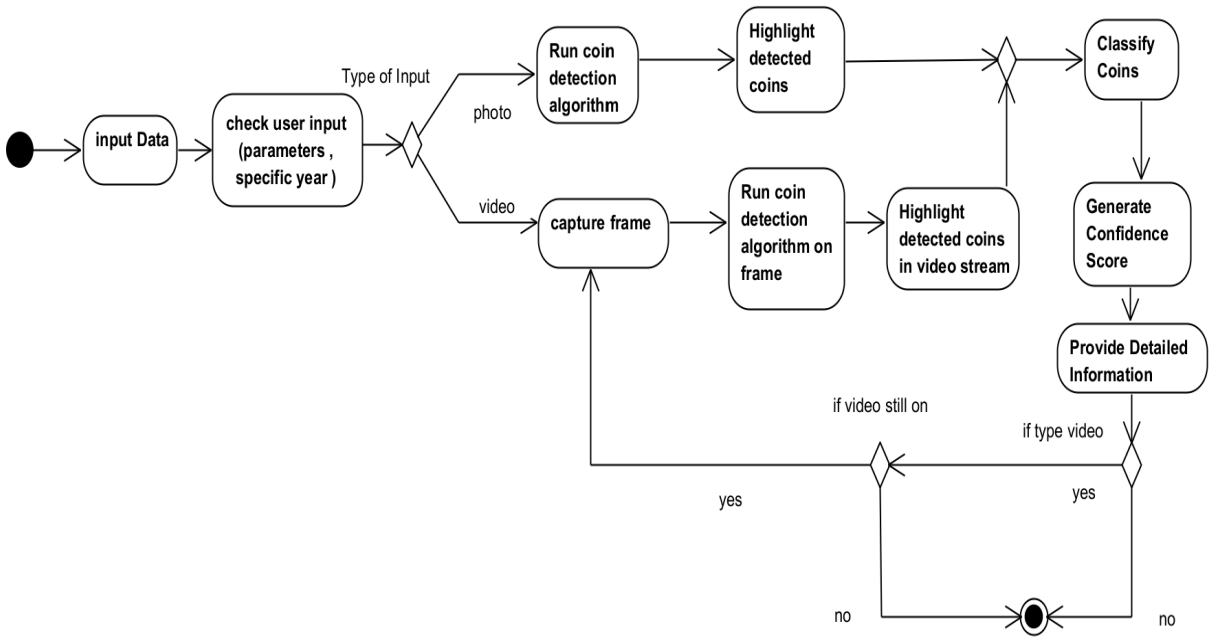

Figure 3: Activity diagram

5.5 Verification and Evaluation

5.5.1 Evaluation

We will evaluate our system based on its ability to accurately identify rare Israeli coins and correctly classify them as either "rare" or "common.", in addition to providing the year of the coin. This classification will rely on the mint year on the coin.

Our primary goal is to develop a fast and accurate detection tool that alerts users to potentially rare coins and provides additional details about their rarity.

Key evaluation criteria include:

- **Precision and Recall:** Assess the model's ability to correctly identify rare coins without excessive false positives or negatives.
- **Real-Time Performance:** Measure the system's ability to detect coins with minimal latency in live video and static image scenarios.

If the system consistently identifies rare coins and gives the correct year of the coin with high precision and recall, it will be considered successful.

5.5.2 Testing Plan

Our iterative development process divides testing into three modules: Object Detection System, Data Analysis, and System Interface. Each module will undergo specific tests to ensure functionality, performance, and accuracy.

Test ID	**Module**	**Tested Function**	**Expected Result**
1	Object Detection System	Image Detection	Coins detected accurately in static images.
2	Object Detection System	Live Video Detection	Real-time detection with steady frame rates (30 fps).
3	Object Detection System	Mint Year Recognition	Accurate identification of mint year from Hebrew inscriptions.
4	Object Detection System	Rare vs. Common Classification	Correctly classifies coins as "rare" or "common" with high accuracy (>80%).
5	Data Analysis	Classification Accuracy	Achieves >85% precision and recall on unseen validation datasets.
6	System Interface	Navigation	Seamless and fast navigation between interface sections.
7	System Interface	Visualization	Displays detection results and summaries.
8	System Interface	Alert System	Alerts users when rare coins are detected in a placer and provides visual indicators of coin placement.

5.5.3 Testing Methods

Unit Testing: Each module will be tested independently to ensure correctness and for object detection, unit tests will focus on coin identification, year extraction, and feature recognition.

Integration Testing: The object detection, data analysis, and system interface will be tested together to validate end-to-end functionality.

Performance Testing: Measure the system's real-time detection speed and ensure it meets performance benchmarks.

User Testing: Simulate real-world scenarios using placers of coins and evaluate usability based on user feedback.

Evaluation Metrics: Precision, Recall, and F1 Score for detection accuracy, latency and framerate for real-time video analysis and user satisfaction ratings for the interface and usability.

By rigorously testing the system using these methods, we aim to deliver a reliable and efficient tool for detecting rare Israeli coins.

5.5.4 GUI

The simple GUI :

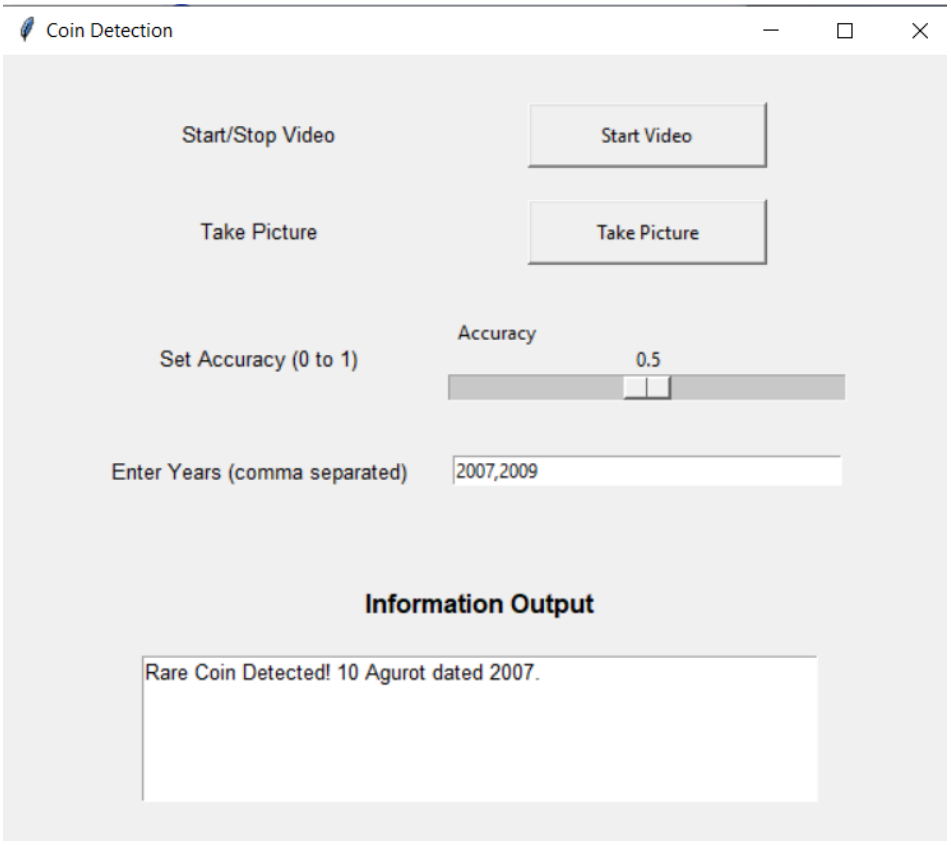

Figure 4: Simple GUI

In addition there will be a picture with the highlighted annotations after the algorithm is ran, for example:

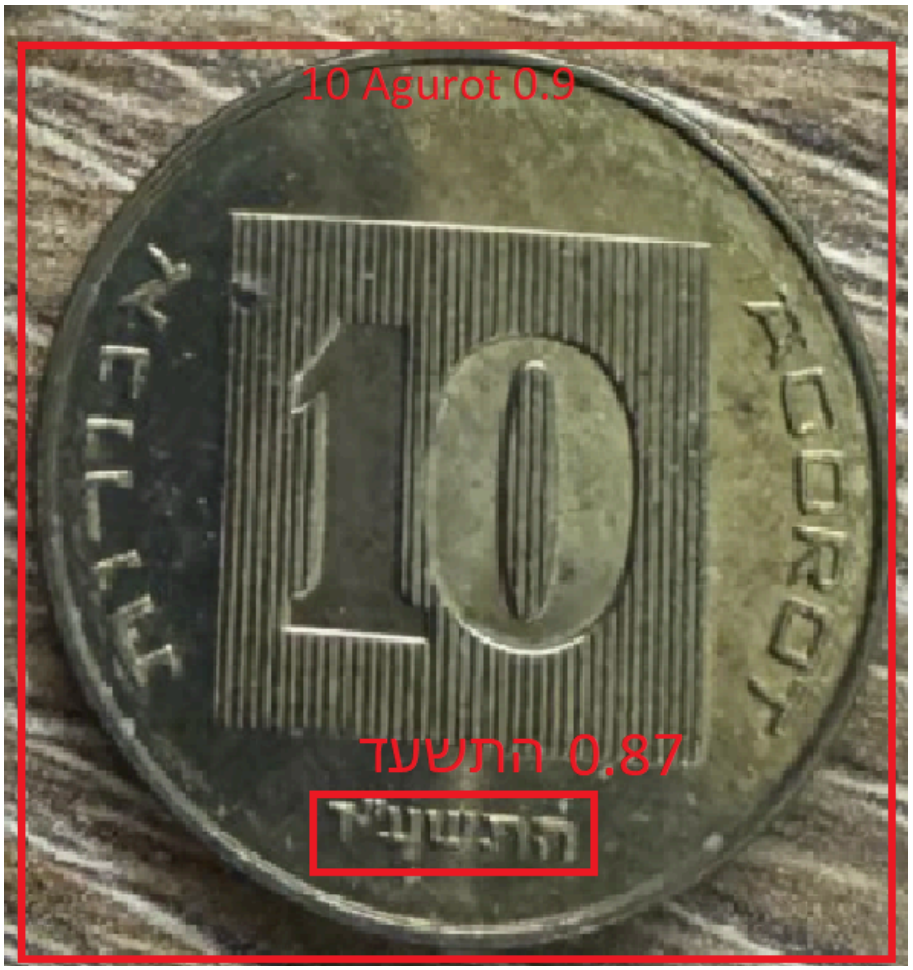

Figure 5: 10 agorot annotated

Figure 6: multiple 10 agorots annotated but only the required date is highlighted in red and a special coin that is found highlighted in blue

6. References

[1] Roomi, S. M. M., & Rajee, R. J. (2015, March). Coin detection and recognition using neural networks. In *2015 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015]* (pp. 1-6). IEEE.

[2] https://www.hebcal.com/home/1824/numerical-values-of-hebrew-letters

[3] Kypraios, I. (Ed.). (2012). *Advances in Object Recognition Systems*. BoD–Books on Demand.

[4] Cooper, J., & Arandjelović, O. (2020). Learning to describe: a new approach to computer vision based ancient coin analysis. *Sci*, *2*(2), 27.

[5] https://en.numista.com/

[6] https://coinoscope.com/

[7] https://pjreddie.com/darknet/yolo/

[8] https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/

[9] https://en.wikipedia.org/wiki/Agile_software_development

[10] https://www.v7labs.com/blog/mean-average-precision

[11] https://docs.ultralytics.com/hub/app/android/